

МИНОБРАЗОВАНИЯ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА



Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

## ОТЧЕТ

по лабораторной работе №2

по дисциплине

Шаблоны проектирования ПО

РУКОВОДИТЕЛЬ:

\_\_\_\_\_

(подпись)

Жевнерчук Д. В.

(фамилия, и.,о.)

СТУДЕНТЫ:

\_\_\_\_\_

(подпись)

Дёмин Д. И.

Карпычева А. Ю.

Пигасин Д. А.

(фамилия, и.,о.)

18-ИВТ-1

(шифр группы)

Работа защищена «\_\_» \_\_\_\_\_

С оценкой \_\_\_\_\_

Нижний Новгород 2019

## Вариант 6

Реализуйте систему генерации шаблона html страницы по описанию:

1. Состав разделов (header, section1, ... , section\_n, footer )
2. Структуры каждой секции (количество блоков div)

Добавьте опциональную возможность выделять определенный раздел, блок рамкой

Сгенерированный html код вводите в консоль.

## Выполнение

Программный код не претерпел особых изменений. Теперь все объекты, которыми манипулирует программа, создаются с помощью Spring Framework.

Описание бинов находится в конфигурационном java классе (src\main\java\app\SpringConfig.java). В ходе работы программы нужные объекты извлекаются из контекста с помощью метода getBean.

Spring берет на себя контроль за жизненным циклом объектов и инъекции зависимостей.

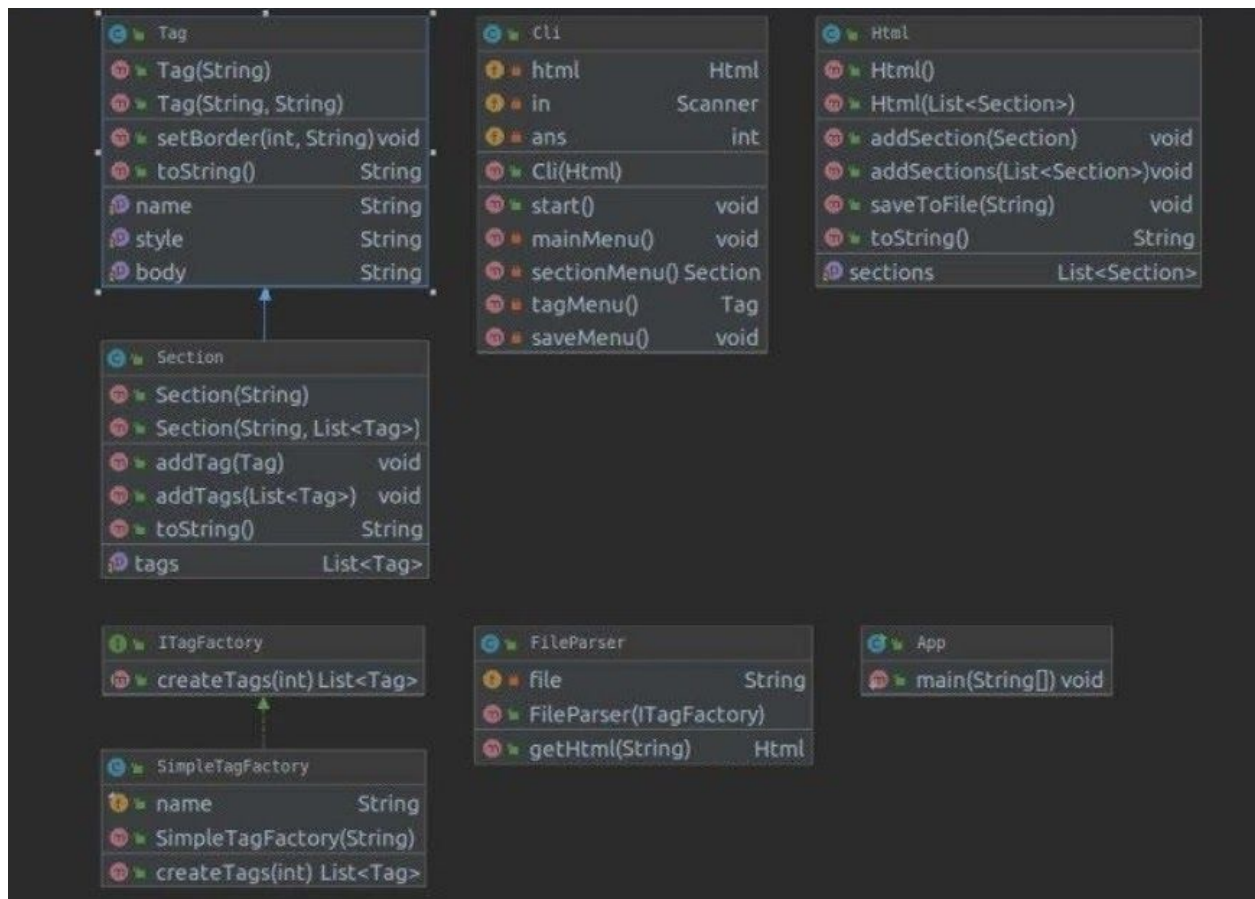
Это облегчает разработку и поддержку крупных проектов.

## SpringConfig.java

```
1  package app;
2
3  import org.springframework.context.annotation.Bean;
4  import org.springframework.beans.factory.annotation.Qualifier;
5  import org.springframework.context.annotation.Configuration;
6
7  import html.FileParser;
8  import html.ITagFactory;
9  import html.SimpleTagFactory;
10 import html.base.Html;
11
12 @Configuration
13 public class SpringConfig {
14
15     ... @Bean
16     ... public FileParser fileParserBean(@Qualifier("divTagFactoryBean") ITagFactory tagFactory) {
17     ...     return new FileParser(tagFactory);
18     ... }
19
20     ... @Bean
21     ... public SimpleTagFactory divTagFactoryBean() {
22     ...     return new SimpleTagFactory("div");
23     ... }
24
25     ... @Bean
26     ... public SimpleTagFactory pTagFactoryBean() {
27     ...     return new SimpleTagFactory("p");
28     ... }
29
30     ... @Bean
31     ... public Html htmlBean(FileParser fp) {
32     ...     return fp.getHtml("html.txt");
33     ... }
34
35     ... @Bean(name = {"cliApp", "cliAppBean"})
36     ... public CliApp cliAppBean(Html html) {
37     ...     return new CliApp(html);
38     ... }
39 }
```

## Реализация

### UML схема



При написании программы мы руководствовались основными принципами ООП, а также использовали паттерны проектирования GOF там, где это уместно.

Например, конструктор класса **FileParser** принимает экземпляр фабрики, реализующей интерфейс **ITagFactory**.

Такой подход позволит нам изменять логику создания тэгов **FileParser**'ом, не переписывая то, что уже было сделано.

Для этого нужно описать новый класс, реализующий интерфейс **ITagFactory** и содержащий нужный нам алгоритм.

## Как использовать

1. Описываем структуру страницы во внешнем файле.
  - Сначала указывается имя секции, количество вложенных блоков.
  - Одна секция - одна строка
2. Запускаем программу
3. Выделяем нужные блоки рамкой (опционально)
4. Сохраняем полученный шаблон (опционально)

## Пример

1. Создаем файл [html.txt](#):

```
html.txt
1  header 10
2  section
3  section 3
4  footer 2
```

2. Запускаем программу.

```
<header>
    <div>div_0</div>
    <div>div_1</div>
    <div>div_2</div>
    <div>div_3</div>
    <div>div_4</div>
    <div>div_5</div>
    <div>div_6</div>
    <div>div_7</div>
    <div>div_8</div>
    <div>div_9</div>
</header>

<section>
</section>

<section>
    <div>div_0</div>
    <div>div_1</div>
    <div>div_2</div>
</section>

<footer>
    <div>div_0</div>
    <div>div_1</div>
</footer>

1.Add section border
2.Add tag border
3.Exit
>>>
```

## Код программы

### **app.App.java**

```
package app;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class App {

    public static void main(String[] args) throws Exception {

        AnnotationConfigApplicationContext ctx = new
        AnnotationConfigApplicationContext(SpringConfig.class);

        CliApp cliApp = ctx.getBean("cliApp", CliApp.class);

        cliApp.start();

        ctx.close();

    }

}
```

### **app.Cli.java**

```
package app;

import java.io.IOException;
import java.util.Scanner;

import html.base.Html;
import html.base.Section;
import html.base.Tag;

public class Cli {

    private Html html;

    private Scanner in = new Scanner(System.in);
```

```
private int ans;
```

```
public Cli(Html html) {  
    this.html = html;  
}
```

```
public void start() throws IOException {  
    do {  
        mainMenu();  
    } while (ans != 3);
```

```
        saveMenu();  
        in.close();  
    }
```

```
private void mainMenu() {  
    System.out.println(html);  
    System.out.println("1.Add section border\n2.Add tag border\n3.Exit");  
    System.out.print(">>>");  
    ans = in.nextInt();
```

```
    if (ans == 1) {  
        Section section = sectionMenu();  
        section.setBorder(5, "black");  
    }
```

```
    if (ans == 2) {  
        Tag tag = tagMenu();  
        tag.setBorder(5, "black");  
    }  
}
```

```

private Section sectionMenu() {
    Section[] sections = html.getSections();

    for (int i = 0; i < sections.length; i++) {
        System.out.println(i + "." + sections[i].getName());
    }

    System.out.print(">>>");
    return sections[in.nextInt()];
}

private Tag tagMenu() {
    Section section = sectionMenu();
    Tag[] tags = section.getTags();

    for (int i = 0; i < tags.length; i++) {
        System.out.println(i + "." + tags[i]);
    }

    System.out.print(">>>");
    return tags[in.nextInt()];
}

private void saveMenu() throws IOException {
    System.out.println("\nSave the file as index.html?\n1.Yes\n2.No");
    ans = in.nextInt();

    if (ans == 1)
        html.saveToFile("index.html");
}

```

```
}
```

### **html.base.Tag.java**

```
package html.base;
```

```
public class Tag {
```

```
    protected String name;
```

```
    protected String body;
```

```
    protected String style;
```

```
    public Tag(String name) {
```

```
        this.name = name;
```

```
    }
```

```
    public Tag(String name, String body) {
```

```
        this.name = name;
```

```
        this.body = body;
```

```
    }
```

```
    public String getName() {
```

```
        return name;
```

```
    }
```

```
    public void setName(String name) {
```

```
        this.name = name;
```

```
    }
```

```
    public String getBody() {
```

```
        return body;
```

```
    }
```



```
public void setBody(String body) {  
    this.body = body;  
}
```

```
public String getStyle() {  
    return style;  
}
```

```
public void setStyle(String style) {  
    this.style = style;  
}
```

```
public void setBorder(int thickness, String color) {  
    setStyle("border:" + thickness + "px solid " + color);  
}
```

```
@Override  
public String toString() {  
    String tagHead = "<" + name;  
    String tagBody = "";  
    String tagTail = "</" + name + ">";
```

```
    if (body != null)  
        tagBody += body;
```

```
    if (style != null)  
        tagHead += " style=" + style + "";
```

```
    tagHead += ">";
```

```

        return tagHead + tagBody + tagTail;
    }

}

html.base.Section.java

package html.base;

import java.util.LinkedList;
import java.util.List;

public class Section extends Tag {
    private List<Tag> tags = new LinkedList<Tag>();

    public Section(String name) {
        super(name);
    }

    public Section(String name, List<Tag> tags) {
        super(name);
        this.tags = tags;
    }

    public Tag[] getTags() {
        return tags.toArray(new Tag[tags.size()]);
    }

    public void setTags(List<Tag> tags) {
        this.tags = tags;
    }

    public void addTag(Tag tag) {

```

```

        tags.add(tag);
    }

    public void addTags(List<Tag> tags) {
        for (Tag tag : tags) {
            addTag(tag);
        }
    }

    @Override
    public String toString() {
        String tagHead = "<" + name;
        String tagBody = "";
        String tagTail = "\n</" + name + ">";

        if (style != null)
            tagHead += " style=\"" + style + "\"";

        tagHead += ">";

        for (Tag tag : tags) {
            tagBody += "\n\t" + tag;
        }

        return tagHead + tagBody + tagTail;
    }
}

```

### **html.base.Html.java**

```
package html.base;
```

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.LinkedList;
import java.util.List;

public class Html {
    private List<Section> sections = new LinkedList<Section>();

    public Html() {
    }

    public Html(List<Section> sections) {
        this.sections = sections;
    }

    public Section[] getSections() {
        return sections.toArray(new Section[sections.size()]);
    }

    public void setSections(List<Section> sections) {
        this.sections = sections;
    }

    public void addSection(Section section) {
        sections.add(section);
    }

    public void addSections(List<Section> sections) {
        for (Section section : sections) {
            addSection(section);
        }
    }
}
```

```
}
```

```
}
```

```
public void saveToFile(String fileName) throws IOException {
```

```
    FileWriter file = new FileWriter(fileName);
```

```
    file.write(toString());
```

```
    file.close();
```

```
}
```

```
@Override
```

```
public String toString() {
```

```
    String html = "";
```

```
    for (Section section : sections) {
```

```
        html += section + "\n\n";
```

```
    }
```

```
    return html;
```

```
}
```

```
}
```

### **html.FileParser.java**

```
package html;
```

```
import java.io.FileNotFoundException;
```

```
import java.io.FileReader;
```

```
import java.util.LinkedList;
```

```
import java.util.List;
```

```
import java.util.Scanner;
```

```
import html.base.Html;
```

```
import html.base.Section;
```

```

public class FileParser {
    private ITagFactory tagFactory;

    public FileParser(ITagFactory tagFactory) {
        this.tagFactory = tagFactory;
    }

    public Html getHtml(String file) throws FileNotFoundException {
        Scanner scan = new Scanner(new FileReader(file));
        List<Section> sections = new LinkedList<Section>();

        while (scan.hasNextLine() & scan.hasNext()) {
            Scanner line = new Scanner(scan.nextLine());
            Section section = new Section(line.next());

            if (line.hasNextInt())
                section.addTags(tagFactory.createTags(line.nextInt()));

            sections.add(section);
        }
        scan.close();

        return new Html(sections);
    }
}

```

### **html.ITagFactory.java**

```

package html;

import java.util.List;

import html.base.Tag;

```

```
public interface ITagFactory {  
    public List<Tag> createTags(int count);  
}
```

### **html.SimpleTagFactory.java**

```
package html;
```

```
import java.util.LinkedList;
```

```
import java.util.List;
```

```
import html.base.Tag;
```

```
public class SimpleTagFactory implements ITagFactory {
```

```
    final public String name;
```

```
    public SimpleTagFactory(String name) {
```

```
        this.name = name;
```

```
    }
```

```
    public List<Tag> createTags(int count) {
```

```
        List<Tag> tags = new LinkedList<Tag>();
```

```
        for (int i = 0; i < count; i++) {
```

```
            tags.add(new Tag(name, name + "_" + i));
```

```
        }
```

```
        return tags;
```

```
    }
```

```
}
```